The Fixed Priority Scheduling Problem for Energy Harvesting Real-Time Systems

Younès Chandarli*†, Yasmina Abdeddaïm*, and Damien Masson*
Université Paris-Est, LIGM UMR CNRS 8049,
*ESIEE Paris, 2 bld Blaise Pascal, BP 99, 93162 Noisy-le-Grand CEDEX, France
†Université Paris-Est Marne-la-vallée, 5 bld Descartes, 77454 Marne-la-Vallée Cedex 2, France

***Abstract*—Energy harvesting is the process of generating electrical energy from environmental sources such as solar panels. In recent years, this term has been frequently applied in the context of small autonomous devices such as wireless sensor nodes. The classical scheduling theory is insufficient for this kind of systems and new scheduling problems arise in this context. Until now, the research on this area focused in trying to improve the efficiency of existing algorithms. Our approach is to complete these efforts by a feasibility theory allowing us to understand why classical optimal algorithms are not efficient anymore with energy constraints.**

In this paper, we try to establish a schedulability test for a fixed priority real-time scheduling problem with energy constraints. We first introduce the problem and describe the model. Then, to illustrate the difficulty of the problem, we focus on a preemptive fixed priority scheduling policy where all the executions are postponed as long as possible. This policy lets the harvester the maximal amount of time to refill the battery. We call this policy PFP_{ALAP} for *As Late As Possible*. We try to define sufficient and/or necessary schedulability conditions and discuss its potential optimality under some additional assumptions. Then, through simple counter examples, we show that intuitive assumptions are wrong for this scheduling problem, making it very interesting to study.

I. Introduction and Related Work

The first work addressing the scheduling problem for energy harvesting systems was the one of Mossé [1]. The problem was solved under a very restrictive task model: the frame based model where all the tasks have exactly the same period and implicit deadline. Then Moser et al. studied a very similar problem [2]. They proposed an optimal algorithm called LSA but in their hypotheses, the CPU frequency can be changed to adjust the Worst Case Execution Time (WCET) of the tasks depending on their energy consumption. Then the results of this work rely on a strong hypothesis: tasks energy consumption is directly linked to their WCET. Recent work shows that this hypothesis is not realistic [3]. Finally, the problem was addressed by Chetto et al. in [4], [5]. A clairvoyant algorithm and several non clairvoyant heuristics was proposed. This clairvoyant algorithm relies on a meta policy: as long as the system can perform without energy failure, a standard policy such as EDF is used. Then, as soon as a future energy issue is detected, i.e $E < 0$, the system is paused as long as possible or until the energy storage unit is full. The notion of slack time [6] is extended to the notion of slack energy. An algorithm to compute slack energy under EDF is provided.

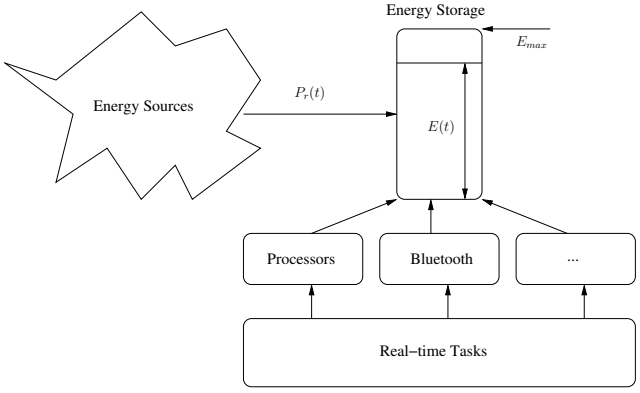

Figure 1. Energy Harvesting Embedded System Model

However, as most embedded systems only use fixed priority, we are interested in extending and completing existing works for fixed priority scheduling.

The remaining of the paper is organized as follow, we present the model in Section II, then we describe the PFP_{ALAP} algorithm and we discuss about its worst case scenario, its study period and we try to build a schedulability condition based on PFP_{ALAP} in Section III. Section IV presents the problem of finding the minimal battery size that permits to schedule a system. Finally we conclude in Section V.

II. Model

A. Target application description

We consider an embedded system connected to an energy harvesting device. An energy harvesting device is a system that collects energy from its environment. The energy is stored in an energy storage unit. We suppose that we can know or bound the energy quantity that arrives in the storage unit, and for each task the energy that it consumes in the worst case. Figure 1 illustrates this description.

B. Formal description

We consider a real-time system in a renewable energy environment defined by a set of n periodic and independent tasks. Each task τ_i is characterized by its priority P_i, its worst case execution time C_i, its period T_i, its deadline D_i and its worst case energy consumption E_i. The execution time C_i and

the energy consumption E_i of a task are fully independent. A task τ_i releases an infinite number of jobs separated by T_i time unit and each job must execute during C_i and consume E_i energy unit. Deadlines are constrained or implicit. The system is powered by a battery or a capacitor recharged from a renewable energy source like a solar panel. The energy is replenished continuously even during the execution of tasks and the energy level of the battery fluctuate between two thresholds E_{max} and E_{min}. We note $Pr(t)$ the charging function and for the sake of simplicity we consider that $P_r(t)$ is a linear function. We define the processor utilization of τ_i as $U_i = C_i/T_i$ and its energy utilization as

$$U_{e_i} = \frac{C_i}{\int_t^{t+T_i} Pr(t)\,\mathrm{d}t}$$

The total utilization of the system is the sum of all the tasks utilization; i.e $U = \sum_1^n U_i$ and $U_e = \sum_1^n U_{e_i}$.

III. PFP_{ALAP} SCHEDULING ALGORITHM

PFP_{ALAP} is a fixed-priority scheduling policy that delays the jobs executions as long as possible, i.e, as long as the slack time remains positive. We are interested in studying this policy since this algorithm is simple and seems optimal when E_{max} is not bounded. It lets the maximal time interval for the battery replenishment before the execution of a task. If this algorithm is proved to be optimal it can be used to build a feasibility test. Unfortunately, we will show a counter example that invalidates this claim. To do that, we must first find the worst case scenario that the task set can suffer, then we have to calculate the length of the study period necessary to decide the task set feasibility. In the following sections, we present without proof an intuitive worst case scenario, a study period and a feasibility condition based on PFP_{ALAP}.

A. PFP_{ALAP} worst case scenario

For the classical work conserving fixed priority algorithm, which we denote PFP_{ASAP} by opposition to PFP_{ALAP}, the worst case activation scenario corresponds to the synchronous activation of all tasks [7]. In this situation the processor gets a maximum of workload that takes a maximum time to execute. So, the resulting busy period is the longest possible one. The lowest priority tasks suffer their longest delay in this situation, this is why it is said the worst case scenario. In the same scenario, PFP_{ALAP} undergoes the same workload and behaves the same way as PFP_{ASAP} but further delays jobs as long as slack-time is available. So without energy constraints, the worst case scenario of this algorithm is the same as PFP_{ASAP}. By combining $E_0 = E_{min}$ with this scenario we get a scenario which can be the worst one for PFP_{ALAP}. Indeed, at this moment the CPU load and energy demand are maximized and the battery level is minimized. A different situation is necessarily not as worse because in the case of an asynchronous activation, we should have more slack-time to recharge and less CPU workload and if more energy than E_{min} is available, the system is less energy constrained. These arguments are persuasive but not conclusive and we do not have any formal proof until now.

B. PFP_{ALAP} Study period

Without energy constraints, the study period of a task set with constrained or implicit deadlines is the processor busy period in the worst case scenario. For PFP_{ASAP} scheduling, this period is the response time of the first job of the lowest priority task [7]. Following the same logic, the study period of PFP_{ALAP} algorithm is defined as the period when there is pending work. It starts from the critical instant (synchronous activation) and ends when all CPU demand is processed, that is later than the effective deadline of the lowest priority task. The first instance may be sufficient to study the schedulability of a system because firstly, all tasks have constrained or implicit deadlines, i.e there is no interference between two successive tasks. Secondly, the next activation period is at least equal or better than a synchronous activation and energy level is $E \geq E_{min}$. This seems reasonable but it is wrong as illustrated in Figure 2. In this counter-example we can see that despite the system described in Table I successes during the lower priority task's response time (time 32), it fails one instance later when the energy level is insufficient to satisfy the energy demand at time 70. To be sure that we will encounter all possible scenarios we must study a longer period bounded by the hyper period. This gives us the possibility to study all possible scenarios. So, the problem of finding a shorter study period is still open.

C. Optimality ?

The aim of our work is to build an energy-aware schedulability test for PFP_{ALAP} that can decide if a given task set is timely and energetically feasible or not, i.e to check if all tasks respect their deadlines while the battery level remains between E_{max} and E_{min}. We begin with setting an assumption for simplicity: we consider initially the battery to be infinite and not bounded, i.e $E_{max} = \infty$, then we relax this hypothesis by fixing a bound for the smallest battery capacity necessary to keep the system feasible.

A schedulability test is the study of tasks execution according to a scheduling policy during a study period that starts at the critical instant in the worst case energy scenario. In this period we have to test if there is no deadline miss nor energy constraints violation. In our case, the algorithm is PFP_{ALAP}, the study period is the hyper period and the worst case scenario is the simultaneous activation at the lowest possible battery level i.e the initial energy level $E_0 = E_{min}$ as previously mentioned. Now it remains to check if all jobs meet their deadlines during the study period, and if the energy recharged during an idle-period is sufficient to meet the energy demand of the busy-period that follows. If the execution satisfies these conditions along the study period, the system is feasible because the periods that follow are at least better or equal to the starting one and the system can execute continuously without constraints violation. To make such verification, we must first calculate the start dates and the length of idle and busy periods and the energy demand of each one.

When the battery reaches its maximum level E_{max} before the end of an idle period, we cannot continue to charge and

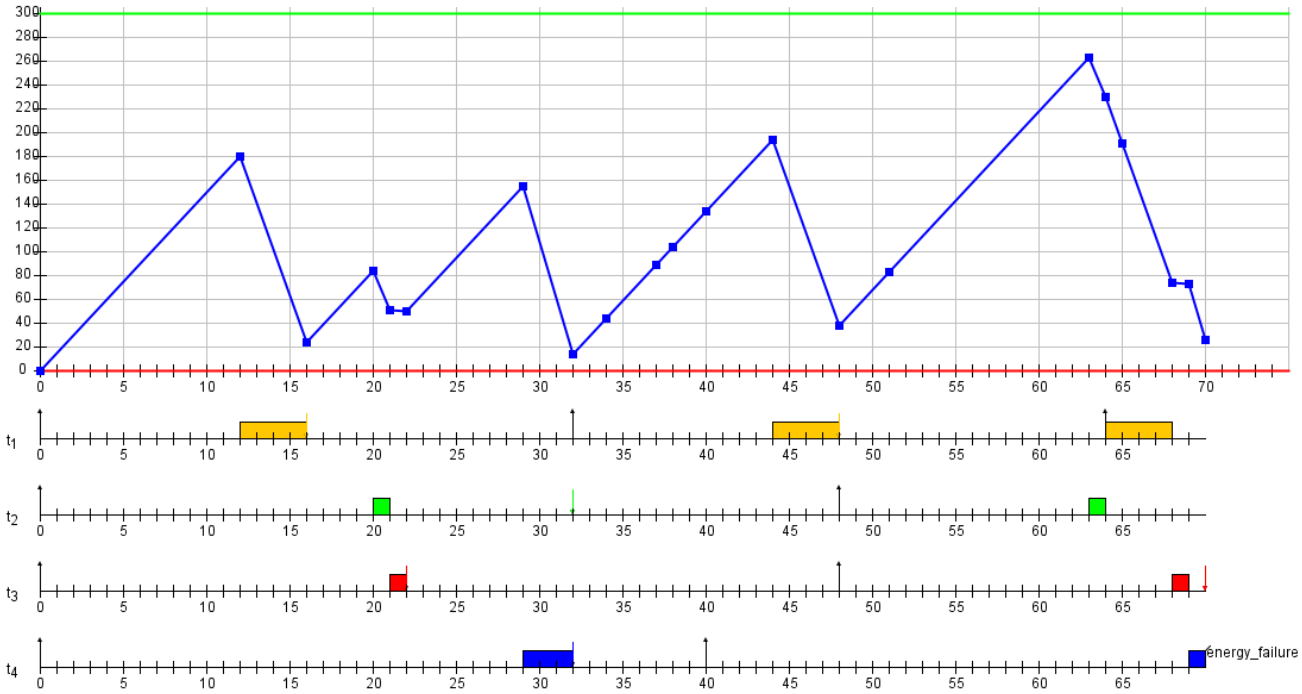

Figure 2. Lowest priority task response time is not the real study period for PFP_{ALAP}

we lose the energy that could be recharged between that moment and the next execution begin. This leads to wast energy if nothing is done to avoid that. In our case this cannot happen because we suppose that E_{max} is infinite. Since PFP_{ALAP} postpones jobs executions to charge battery as much as possible before executing, our intuition is to say that such scheduling policy is optimal. We know that all schedulable task sets with PFP_{ASAP} are schedulable with PFP_{ALAP} so no timing problems can occur and we know also that in PFP_{ALAP} idle-periods are as longer as possible which seems better for energy level. These simple intuitions may appear convincing but when we tried to build a formal proof we found a counter-example which invalidates the optimality of PFP_{ALAP}. Figures 3 and 2 illustrate a counter example where the task set described in Table I is not schedulable with PFP_{ALAP} while it exists a feasible schedule. Indeed, we simulated this system with YARTISS [8]; and this simulation shows that it is feasible until the hyper period. We can see that PFP_{ALAP} postpones executions as late as possible which leads in this example to a very long busy period which has a big energy demand and the energy charged during the previous idle period is not sufficient, thus the system failed energetically. When we schedule the same task set with PFP_{ASAP} by introducing charging periods as much as necessary when the battery is empty, jobs are scheduled as soon as possible then we avoid some very long busy periods and all jobs meet their deadline and idle periods are always sufficient.

-	C_i	E_i	T_i	D_i	P_i
τ_1	4	216	32	16	1
τ_2	1	48	48	32	2
τ_3	1	16	48	22	3
τ_4	3	186	40	32	4

Table I
CONTER-EXAMPLE : WITH $E_{max} = 300$, $E_{min} = 0$ AND $P_r(t) = 15$

IV. BATTERY CAPACITY

The design of a system with $E_{max} = \infty$ assumption is not realistic and schedulability test based on such assumption is not very useful, so finding the lowest E_{max} value is necessary. The exact capacity of the battery is difficult to compute because it depends on tasks characteristics and we cannot predict the lowest necessary capacity to satisfy energy demand continuously before executing tasks. So to prevent this, we can solve the problem partially by bounding E_{max} value.

First, knowing that tasks have implicit or constrained deadlines we can limit the study period to the hyper period. Second, the CPU utilization U gives us the proportion of time where CPU is busy, the remaining time the processor is idle and the system is only charging energy. So the highest level of energy that can be reached during a cycle starting from the worst case

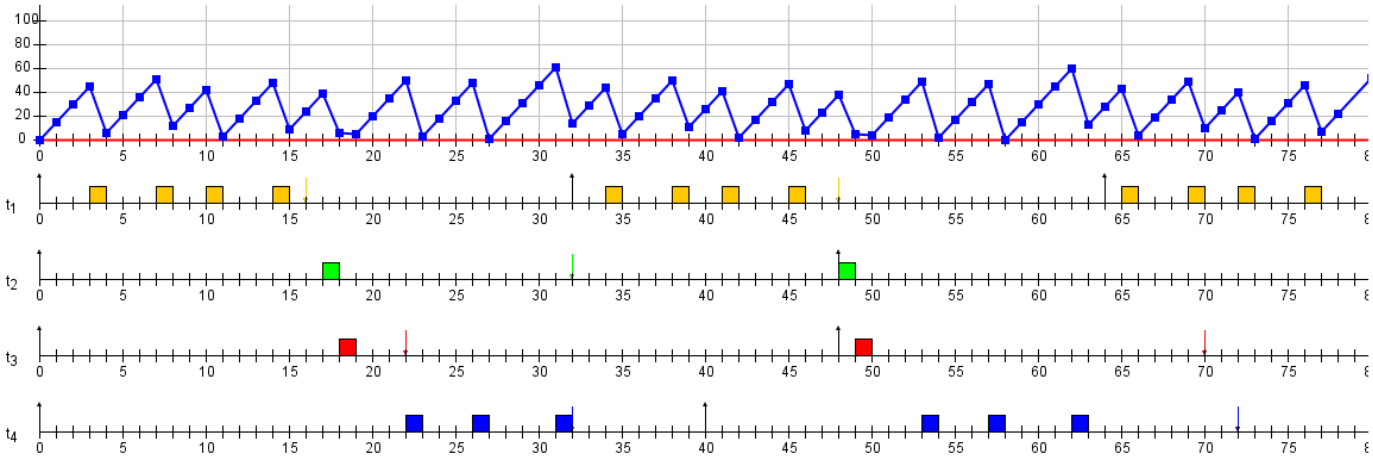

Figure 3. PFP_{ALAP} is not optimal

scenario of activation and energy is

$$Bound_{max} = \int_0^{(1-U)\times HyperPeriod} Pr(t)\,\mathrm{d}t.$$

This is because we suppose for all tasks that it consumes more energy than we can charge during its execution time, i.e $E_i > \int_0^{C_i} Pr(t)\,\mathrm{d}t$. This bound is very pessimistic but gives us a sufficient condition. If a task set is schedulable with PFP_{ALAP} with $E_{max} = \infty$ and the given E_{max} is greater than or equal to $Bound_{max}$, then it is feasible. By combining the previous test with a battery capacity bound, we have a sufficient condition for the feasibility of a task set.

V. Conclusion and Future works

In this paper we presented some intuitive ideas to solve the problem of the energy harvesting scheduling. Then we showed that simple solutions are not always the best choices despite their obviousness. We invalidated the optimality of PFP_{ALAP} and showed that the classical study period is no more valid. We also presented a sufficient schedulability test based on PFP_{ALAP} algorithm and proposed a bound for the lowest necessary battery capacity. It is not an optimal algorithm, so a such schedulability test may be very pessimistic. For this reason we will continue our research on this way by exploring other scheduling policies and heuristics and try to provide more optimistic feasibility conditions. We will study carefully PFP_{ASAP} and other clairvoyant algorithms and try to establish more optimistic or optimal feasibility conditions.

References

[1] A. Allavena and D. Mossé, "Scheduling for frame-based embedded systems with rechargeable batteries," *In Workshop on Power Management for Real-Time and Embedded Systems (in conjunction with RTAS)*, 2001.

[2] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Lazy scheduling for energy harvesting sensor nodes," *5th IFIP Working Conference on Distributed and Parallel Embedded Systems DIPES 2006, Braga, 11-13 Oct 06*, 2006.

[3] R. Jayaseelan, T. Mitra, and X. Li, "Estimating the worst-case energy consumption of embedded software," in *IEEE Real Time Technology and Applications Symposium*, 2006, pp. 81–90.

[4] M. Chetto, H. E. Ghor, and R. H. Chehade, "A real-time scheduling framework for embedded systems with environmental energy harvesting," *Computers and Electrical Engineering archive Volume 37 Issue 4, July, 2011*, 2011.

[5] M. Chetto, D. Masson, and S. Midonnet, "Fixed priority scheduling strategies for ambient energy-harvesting embedded systems," *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference*, 2011.

[6] J. P. Lehoczky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks fixed priority preemptive systems," in *proceedings of the 13th IEEE Real-Time Systems Symposium*, Phoenix, Arizona, Dec. 1992, pp. 110–123.

[7] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973. [Online]. Available: http://doi.acm.org/10.1145/321738.321743

[8] Y. Chandarli, F. Fauberteau, D. Masson, S. Midonnet, and M. Qamhieh, "Yartiss: A tool to visualize, test, compare and evaluate real-time scheduling algorithms," *3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, 2012.